This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-207854
(43)Date of publication of application : 07.08.1998

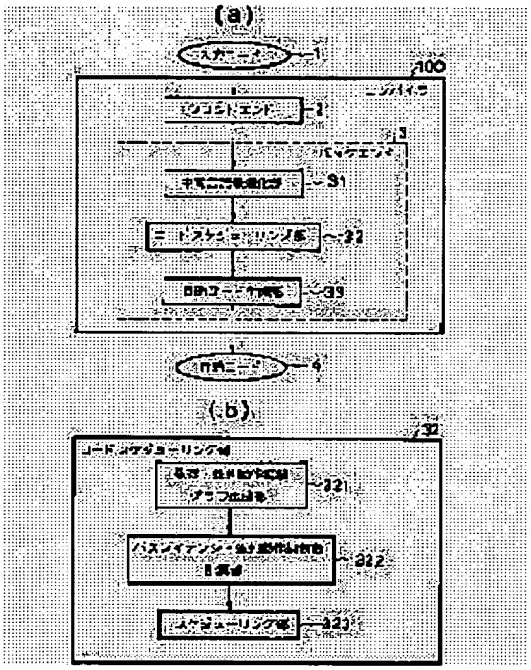
(51)Int.Cl. G06F 15/16
G06F 9/45

(21)Application number : 09-024362
(22)Date of filing : 23.01.1997

(71)Applicant : NEC CORP
(72)Inventor : MIYAMOTO TAKASHI

(54) COMPILER INSTRUCTION PARALLELIZING SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To generate a high-speed object code by increasing the parallelism of the object code (object code for a computer which has arithmetic units capable of operating in parallel).
SOLUTION: A dependency and parallel operation suppression graph generation part 321 generates a dependency and parallel operation suppression graph as a directional graph representing dependency relation and parallel operation suppression relation between instructions. A path latency and parallel operation suppression number calculation part 322 calculates a path latency and parallel operation suppression number on the basis of the dependency and parallel operation suppression graph generated by the dependency and parallel operation suppression graph generation part 321 and adds the calculation result to the dependency and parallel operation suppression graph. A scheduling part 323 schedules codes in parallel according to the dependency and parallel operation suppression graph to which the calculation result of the path latency and parallel operation suppression number is added.



LEGAL STATUS

[Date of request for examination] 23.01.1997
[Date of sending the examiner's decision of rejection] 24.11.1999
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－207854

(43)公開日　平成10年(1998)８月７日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　15/16	４３０	Ｇ０６Ｆ　15/16	４３０Ａ
9/45		9/44	３２２Ｆ

審査請求　有　　請求項の数４　ＦＤ　（全９頁）

(21)出願番号　特願平9－24362

(22)出願日　平成９年(1997)１月23日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　宮本　敬士
東京都港区芝五丁目７番１号　日本電気株式会社内

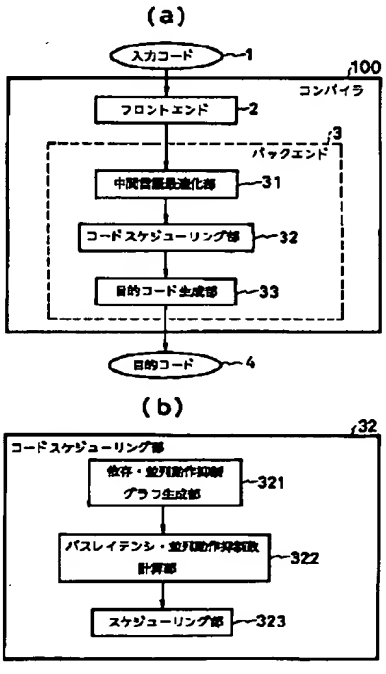
(74)代理人　弁理士　河原　純一

(54)【発明の名称】　コンパイラ命令並列化方式

(57)【要約】

【課題】　目的コード（並列に動作し得る複数の演算ユニットを持つ計算機用の目的コード）の並列性を高め、高速な目的コードを生成する。

【解決手段】　依存・並列動作抑制グラフ生成部３２１は、命令間の依存関係および並列動作抑制関係を表現する有向グラフである依存・並列動作抑制グラフを生成する。パスレイテンシ・並列動作抑制数計算部３２２は、依存・並列動作抑制グラフ生成部３２１により生成される依存・並列動作抑制グラフに基づいてパスレイテンシおよび並列動作抑制数を計算し、その計算結果を依存・並列動作抑制グラフに追加する。スケジューリング部３２３は、パスレイテンシ・並列動作抑制数計算部３２２によりパスレイテンシおよび並列動作抑制数の計算結果が追加された依存・並列動作抑制グラフに基づいて、コードの並列なスケジューリングを行う。



1

【特許請求の範囲】

【請求項１】　命令間の依存関係および並列動作抑制関係を表現する有向グラフである依存・並列動作抑制グラフを生成する依存・並列動作抑制グラフ生成部と、

前記依存・並列動作抑制グラフ生成部により生成される依存・並列動作抑制グラフに基づいてパスレイテンシおよび並列動作抑制数を計算し、その計算結果を依存・並列動作抑制グラフに追加するパスレイテンシ・並列動作抑制数計算部と、

前記パスレイテンシ・並列動作抑制数計算部によりパスレイテンシおよび並列動作抑制数の計算結果が追加された依存・並列動作抑制グラフに基づいてコードの並列なスケジューリングを行うスケジューリング部とを有することを特徴とするコンパイラ命令並列化方式。

【請求項２】　前記依存・並列動作抑制グラフ生成部により生成された依存・並列動作抑制グラフに対して、有向木のルートから有向辺に従ってノードを辿ることにより各命令のパスレイテンシを決定し、各ノードから出ている無向辺の数を計算することにより並列動作抑制数を決定する前記パスレイテンシ・並列動作抑制数計算部と、

最初にパスレイテンシが大きい命令から順にスケジューリングし、同じパスレイテンシの命令については並列動作抑制数が大きい命令から順にスケジューリングする前記スケジューリング部とを有することを特徴とする請求項１記載のコンパイラ命令並列化方式。

【請求項３】　中間言語コードに対してスケジューリングを行う前記スケジューリング部を有することを特徴とする請求項１または請求項２記載のコンパイラ命令並列化方式。

【請求項４】　スケジュール前目的コードに対してスケジューリングを行う前記スケジューリング部を有することを特徴とする請求項１または請求項２記載のコンパイラ命令並列化方式。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンパイラ命令並列化方式に関し、特に並列に動作し得る複数の演算ユニットを持つ計算機（コンピュータ）用の目的コードを生成するコンパイラにおけるコンパイラ命令並列化方式に関する。

【０００２】

【従来の技術】従来より、この種のコンパイラ命令並列化方式は、上記のような計算機（複数命令を並列に実行可能な計算機であり、同時に動作し得る複数の演算ユニットを持つ計算機）用の目的コードの効率を高めるために用いられている。

【０００３】従来のこの種のコンパイラ命令並列化方式としては、例えば、以下の①～③に示す特許公報に記載されたものがあった。

2

【０００４】①　特開平６－２５０８４７号公報には、並列論理型言語の命令をＶＬＩＷ（Ｖｅｒｙ　Ｌｏｎｇ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｗｏｒｄ）方式の並列計算機で実行できるようなＶＬＩＷ命令に変換する技術が記載されている。この公報に記載されたコンパイラ命令並列化方式では、並列論理型言語がコンパイラにより抽象命令列に変換された後に、並列論理型言語の特徴が生かされて並列にスケジューリングされ、さらに命令間の依存関係に基づいて並列にスケジューリングが行われる。

【０００５】②　特開平６－２９５２４６号公報には、後向きコード圧縮フェーズによりスケジュールを行うことによって、レジスタの使用量を削減してシステムの処理の遅延を防止し、処理の効率化を図る技術が記載されている。この公報に記載されたコンパイラ命令並列化方式では、前向きリストスケジューリングという、依存関係に基づいて並列にスケジューリングを行う方式が適用された後に、後向きコード圧縮フェーズにより命令が移動されて、レジスタの使用量の削減が行われる。

【０００６】③　特開平６－１３１１９５号公報には、既存マシン用に作成されたアセンブルテキストが表現する命令列中の、各命令の依存関係を解析する命令解析手段による解析結果に基づいて、命令の並べ替えを行う技術が記載されている。この公報に記載されたコンパイラ命令並列化方式では、アセンブルテキストが表す命令列をもとに命令依存グラフが作成されることによって命令間の依存関係が解析され、この結果に基づいて並列にスケジューリングが行われる。

【０００７】

【発明が解決しようとする課題】上述した従来のコンパイラ命令並列化方式には、複数命令を並列に実行することが可能であり同時に動作し得る複数の演算ユニットを持つ計算機（ある種のマイクロプロセッサ等）において、依存関係（ある命令が書き込んだレジスタを他の命令で読み出すなどの原因により発生する、命令間の実行順序の関係）および資源競合（同一の演算ユニットを複数の命令が使用できないこと）以外の原因により発生する命令間の並列動作が不可能になる関係が生じる場合に、必ずしも目的コードの最適なスケジューリングを行うことができないという問題点があった。

【０００８】このような問題点が生じる理由は、目的コードのスケジューリングが命令間の依存関係および資源競合のみに基づいて行なわれているためである。

【０００９】なお、依存関係および資源競合以外の原因により発生する命令間の並列動作が不可能になるという関係（以下、これを「並列動作抑制関係」という）は、本発明の属する技術分野に含まれるマイクロプロセッサではしばしば発生するものである。例えば、２個の演算ユニットを持つペンティアム（Ｐｅｎｔｉｕｍ）プロセッサ（アメリカ合衆国インテル（Ｉｎｔｅｌ）社製）では、他命令と並列に実行できない命令や、特定の演算ユ

ニットでその命令を実行する時に限って他命令と並列に実行できるという命令がある。これらの命令における制限は、命令間の依存関係や資源競合では扱うことができない。

【００１０】本発明の目的は、上述の点に鑑み、並列に動作し得る複数の演算ユニットを持つ計算機用の目的コードを生成するコンパイラにおいて、目的コードの並列性を高めることにより高速な目的コードを生成することができるコンパイラ命令並列化方式を提供することにある。

【００１１】

【課題を解決するための手段】本発明のコンパイラ命令並列化方式は、命令間の依存関係および並列動作抑制関係を表現する有向グラフである依存・並列動作抑制グラフを生成する依存・並列動作抑制グラフ生成部と、前記依存・並列動作抑制グラフ生成部により生成される依存・並列動作抑制グラフに基づいてパスレイテンシおよび並列動作抑制数を計算し、その計算結果を依存・並列動作抑制グラフに追加するパスレイテンシ・並列動作抑制数計算部と、前記パスレイテンシ・並列動作抑制数計算部によりパスレイテンシおよび並列動作抑制数の計算結果が追加された依存・並列動作抑制グラフに基づいてコードの並列なスケジューリングを行うスケジューリング部とを有する。

【００１２】なお、本発明のコンパイラ命令並列化方式は、特に、以下の①および②に示す構成要素を有するように構成することができる。

【００１３】①　前記依存・並列動作抑制グラフ生成部により生成された依存・並列動作抑制グラフに対して、有向木のルートから有向辺に従ってノードを辿ることにより各命令のパスレイテンシを決定し、各ノードから出ている無向辺の数を計算することにより並列動作抑制数を決定する前記パスレイテンシ・並列動作抑制数計算部

【００１４】②　最初にパスレイテンシが大きい命令から順にスケジューリングし、同じパスレイテンシの命令については並列動作抑制数が大きい命令から順にスケジューリングする前記スケジューリング部

【００１５】

【発明の実施の形態】次に、本発明の実施の形態について、図面を参照して詳細に説明する。

【００１６】

【実施例】

（１）　本発明の第１の実施例

図１（ａ）および（ｂ）は、本発明のコンパイラ命令並列化方式の第１の実施例の構成を示すブロック図である。

【００１７】図１（ａ）に示すように、本実施例のコンパイラ命令並列化方式は、フロントエンド２と、バックエンド３とを含むコンパイラ１００によって実現される。コンパイラ１００は、入力コード１を入力とし、目

的コード４を出力とする。

【００１８】フロントエンド２は、入力コード１であるプログラムの字句解析および構文・意味解析を行い、中間言語コードを出力する。

【００１９】バックエンド３は、中間言語最適化部３１と、コードスケジューリング部３２と、目的コード生成部３３とを含む。

【００２０】中間言語最適化部３１は、中間言語コードに対して広域最適化および局所最適化を行う。

【００２１】コードスケジューリング部３２は、中間言語コードの命令を並列実行性が高くなるように並べ替える。このコードスケジューリング部３２が、本発明の主要部に該当する。

【００２２】目的コード生成部３３は、中間言語コードを目的コード４に変換する。

【００２３】次に、図１（ｂ）を参照して、コードスケジューリング部３２の詳細な構成を説明する。

【００２４】コードスケジューリング部３２は、依存・並列動作抑制グラフ生成部３２１と、パスレイテンシ・並列動作抑制数計算部３２２と、スケジューリング部３２３とを含む。

【００２５】依存・並列動作抑制グラフ生成部３２１は、中間言語コードを構成する命令間の依存性を解析し、この解析結果に基づき有向グラフを構築する。さらに、依存・並列動作抑制グラフ生成部３２１は、並列動作抑制関係も解析し、先に構築した有向グラフに並列動作抑制関係を追加する。このようにして構築された有向グラフを依存・並列動作抑制グラフと呼ぶ。

【００２６】パスレイテンシ・並列動作抑制数計算部３２２は、依存・並列動作抑制グラフを用いて、パスレイテンシと並列動作抑制数とを計算する。

【００２７】スケジューリング部３２３は、依存・並列動作抑制グラフとパスレイテンシおよび並列動作抑制数とを用いて、中間言語コードの命令の並べ替えを行う。

【００２８】図２は、コードスケジューリング部３２の処理を示す流れ図である。この処理は、依存関係有向グラフ表現ステップ２０１と、並列動作抑制関係解析ステップ２０２と、パスレイテンシ計算ステップ２０３と、並列動作抑制数計算ステップ２０４と、中間言語コード命令並べ替えステップ２０５とからなる。

【００２９】図３は、依存・並列動作抑制グラフ生成部３２１が生成する有向グラフ（依存・並列動作抑制グラフ）を説明するための図である。

【００３０】図４は、スケジューリング部３２３によるスケジューリング結果である出力コードの一例を示す図である。

【００３１】図５は、本実施例のコンパイラ命令並列化方式の動作を説明するための図であり、本発明のコンパイラ命令並列化方式を使用しないと仮定した時のスケジューリング結果である出力コードの一例を示す図であ

る。

【００３２】次に、このように構成された本実施例のコンパイラ命令並列化方式の動作について説明する。

【００３３】第１に、図１（ａ）に示すコンパイラ１００の概略的な動作について説明する。

【００３４】コンパイラ１００内のフロントエンド２は、入力コード１を中間言語コードに変換する。この中間言語コードは、バックエンド３に渡される。

【００３５】バックエンド３においては、まず、中間言語最適化部３１が、中間言語コードに対して局所最適化および広域最適化を行う。

【００３６】次に、コードスケジューリング部３２は、並列実行性を高めるための命令の並べ替えを行う。

【００３７】さらに、目的コード生成部３３は、中間言語コードを目的コード４に変換し、コンパイラ１００の出力コードである目的コード４を生成する。

【００３８】第２に、図１（ａ）および（ｂ）と共に図２，図３，および図４を参照して、本発明の中心となるコードスケジューリング部３２における動作について説明する。

【００３９】コードスケジューリング部３２では、最初に依存・並列動作抑制グラフ生成部３２１が、依存・並列動作抑制グラフを生成する。

【００４０】すなわち、依存・並列動作抑制グラフ生成部３２１は、依存・並列動作抑制グラフを生成するために、まず命令間の依存関係を解析し、その依存関係を有向グラフ（初期の依存・並列動作抑制グラフ）で表現する（ステップ２０１）。

【００４１】図３に示された依存・並列動作抑制グラフの一例により、依存・並列動作抑制グラフによって表現される依存関係を説明する。依存・並列動作抑制グラフでは、中間言語コードの各命令は有向グラフのノードで表現される。図３中の「ａ」および「Ｂ」等は命令を表すノードである。また、依存・並列動作抑制グラフでは、依存関係は有向グラフの有向辺で表現される。図３では、例えば命令Ｂは命令ａに依存しているので、ノードａからノードＢへの有向辺が存在する。

【００４２】次に、依存・並列動作抑制グラフ生成部３２１は、依存・並列動作抑制グラフを生成するために、並列動作抑制関係を解析し、その解析結果を依存・並列動作抑制グラフに追加する（ステップ２０２）。

【００４３】並列動作抑制関係とは、先に述べたように、依存関係および資源競合以外の原因により２個の命令の並列動作が不可能となる関係をいう。一例としては、「パイプライン動作で一部共通回路を使用することにより、２個の命令が別々の演算ユニットで実行されるにもかかわらず、並列実行できないという関係」が考えられる。並列動作抑制関係は、依存・並列動作抑制グラフにおいて無向辺で表現される。再び図３で説明すると、例えば命令Ｂと命令ｅとは並列動作抑制関係にあるので、ノードＢとノードｅとは無向辺で結ばれている。

【００４４】パスレイテンシ・並列動作抑制数計算部３２２は、まず、依存・並列動作抑制グラフに基づいてパスレイテンシを計算し、その計算結果を依存・並列動作抑制グラフに追加する（ステップ２０３）。

【００４５】パスレイテンシとは、依存・並列動作抑制グラフにおいて、各命令が最短で実行可能となるサイクル数（各ノードについて、そのノードが含まれる有向木のルートに該当する命令からの最短実行可能サイクル数）をいう。以後説明の都合上全ての命令の実行に要するサイクル数は１であると仮定しているが、これ以外の場合でも本発明のコンパイラ命令並列化方式が適用可能であることはいうまでもない。

【００４６】図３を参照すると、例えば命令Ｂは命令ａに依存しているので、命令ａの実行が完了しないと命令Ｂが実行可能とならない。命令ａは依存している命令がないので、この依存・並列動作抑制グラフでは最初に実行可能であり、この場合には、命令ａのパスレイテンシとして０が与えられる。前述の仮定により命令ａの実行に要するサイクル数は１であるから、命令Ｂのパスレイテンシは１となる。図３では、パスレイテンシを、各ノードの近くにおけるカッコで囲まれない数で表している。

【００４７】続いて、パスレイテンシ・並列動作抑制数計算部３２２は、依存・並列動作抑制グラフに基づいて並列動作抑制数を計算し、その計算結果を依存・並列動作抑制グラフに追加する（ステップ２０４）。

【００４８】並列動作抑制数とは、各命令について、その命令と並列動作抑制関係にある命令の数をいう。したがって、依存・並列動作抑制グラフでは、命令に対応するノードから出ている無向辺の数が並列動作抑制数に該当する。図３を参照すると、例えば命令Ｂの並列動作抑制数は１である。図３では、並列動作抑制数を、各ノードの近くにおけるカッコで囲まれた数で表している。ただし、カッコで囲まれた数が存在しない命令（ノード）については、その命令の並列動作抑制数は０である。

【００４９】スケジューリング部３３３は、以上のようにして最終的に構築された依存・並列動作抑制グラフに基づいて、中間言語コードの命令の並べ替え（所定のスケジューリング）を行う（ステップ２０５）。

【００５０】スケジューリング部３３３の動作について、図３と図４とを参照して説明する。説明の都合上、目的コード４は、並列に動作する２個の演算ユニット（これらを演算ユニット①および演算ユニット②とする）を持つ計算機向けの目的コードであると仮定する。また、命令が実行される演算ユニットは命令毎に決まっており、図３の依存・並列動作抑制グラフに含まれる命令については、命令ａ，ｅ，ｆ，およびｈが演算ユニット①で実行され、命令Ｂ，Ｃ，Ｄ，およびＧが演算ユニット②で実行されるものと仮定する。

7

【００５１】本発明では、並列なスケジューリングの方法として、従来の技術であるリストスケジューリングを基本とするが、本発明の特徴である並列動作抑制数を考慮したスケジューリングが行われる。リストスケジューリングでは、依存・並列動作抑制グラフにおいて、スケジューリング可能な命令（依存されている命令でスケジューリングが未了なものがないような命令）の中でパスレイテンシが最大の命令がスケジューリングの候補とされ、逆向きに（後に実行される命令から順に）スケジューリングされた命令列が決定される。

【００５２】図４は、本実施例のコンパイラ命令並列化方式によってスケジューリングされた命令列を示す図である。図４の命令列は、最初のクロックで命令ａと命令Ｄとが実行され、次のクロックで命令ｆと命令Ｇとが実行され、さらに以下同様に各命令が実行されることを想定してスケジューリングされた命令列である。このような命令列を決定するにあたっては、前述のリストスケジューリングの方法に従い、後に実行される命令、すなわち命令ｅと命令Ｃとから始め、逆向きにスケジューリングされることになる。

【００５３】図３に戻り、図４に示す命令列がスケジューリングされる過程を示す。最初は、どの命令もスケジューリングされていないため、スケジューリングの候補となり得る命令は、命令Ｃ，ｅ，ｆ，およびｈ（有向辺の開始ノードとなっていない命令）となる。このうちパスレイテンシが最大なものは命令Ｃであるので、まずこの命令Ｃがスケジュール（演算ユニット②に対するスケジュール）される。残りの３個の命令ｅ，ｆ，およびｈは全てパスレイテンシが１であり、また全て演算ユニット①で実行されるので、命令Ｃと資源競合を起こさないため、通常のリストスケジューリングの方法でこれらの命令の中から任意に１個の命令が選択されることになる。

【００５４】本発明の方法では、このようにパスレイテンシが等しく、既にスケジュール済みの命令と資源競合を起こさない命令がスケジュール候補となった場合には、並列動作抑制数が最大な命令が選択される。ただし、既にスケジュール済みの命令と並列動作抑制関係となる命令は除かれる。図３では、命令ｅの並列動作抑制数が最大であるため、演算ユニット①にはこの命令ｅが実行されるようにスケジュールされる。

【００５５】以下同様にして、図４に示すスケジューリング結果を得るためのスケジューリングが行われる。

【００５６】図５に、本発明の方法によらず、演算ユニット①で命令ｅ，ｆ，およびｈの３個の命令から１個の命令が選択される時点で、命令ｈが選択されるとした時の残りのスケジューリング結果を示す。本発明の方法でスケジュールした結果である図４のスケジューリング結果に比べると、明らかに実行サイクル数が多くなっており、生成される目的コードの性能が低下していることが

8

分かる。

【００５７】（２）　本発明の第２の実施例

図６は、本発明のコンパイラ命令並列化方式の第２の実施例の構成を示すブロック図である。なお、コードスケジューリング部７の構成は、図１（ｂ）に示すコードスケジューリング部３２の構成と同様である。

【００５８】本実施例のコンパイラ命令並列化方式は、フロントエンド２と、バックエンド５と、スケジュール前目的コード６と、コードスケジューリング部７とを含むコンパイラ２００によって実現される。コンパイラ２００は、入力コード１を入力とし、目的コード４を出力とする。

【００５９】フロントエンド２は、入力コード１であるプログラムの字句解析および構文・意味解析を行い、中間言語コードを出力する。

【００６０】バックエンド５は、中間言語最適化部３１と、目的コード生成部３３とを含み、中間言語コードに基づいてスケジュール前目的コード６を出力する。

【００６１】コードスケジューリング部７は、スケジュール前目的コード６の命令を並列実行性が高くなるように並べ替える。

【００６２】図６に示されるように、コードスケジューリング部７は、図１に示された第１の実施例におけるバックエンド３（本実施例ではバックエンド５）に含まれず、バックエンド５によって出力されるスケジュール前目的コード６を入力する位置に存在する。この点で、第２の実施例の構成は第１の実施例の構成と異なる。

【００６３】次に、このように構成された本実施例のコンパイラ命令並列化方式の動作について説明する。

【００６４】第１の実施例においては、コードスケジューリング部３２は中間言語最適化部３１が出力する中間言語コードに対してコードスケジューリング（上述したような本発明に特有のコードスケジューリング）を行っていた。これに対して、本実施例（第２の実施例）のコンパイラ命令並列化方式においては、コードスケジューリング部７はバックエンド５が出力するスケジュール前目的コード６に対して所定のコードスケジューリングを行う。その他の動作については、第１の実施例と第２の実施例とにおいて、同じである。

【００６５】

【発明の効果】以上説明したように、本発明によると、並列動作抑制関係を反映させた依存・並列動作抑制グラフを構築すると共に、パスレイテンシおよび並列動作抑制数を計算することによって、並列コードスケジューリングの効率が向上し、目的コードの実行性能が向上するという効果が生じる。

【００６６】このような効果が生じる理由は、依存・並列動作抑制グラフに基づいて並列動作抑制数の大きい命令を先にスケジュールすることによって、スケジューリング処理の後の段階で並列動作抑制関係によるスケジュ

9

ーリング制約が発生するのを防ぐことができるためである。

【図面の簡単な説明】

【図１】本発明のコンパイラ命令並列化方式の第１の実施例の構成を示すブロック図である（特に、（ｂ）はコードスケジューリング部の詳細な構成を示すブロック図である）。

【図２】図１（ｂ）に示すコードスケジューリング部の処理を示す流れ図である。

【図３】図１（ｂ）に示すコードスケジューリング部が生成する有向グラフ（依存・並列動作抑制グラフ）の一例を示す図である。

【図４】図１（ｂ）中のスケジューリング部によるスケジューリング結果である出力コードの一例を示す図である。

【図５】図１（ａ）に示すコンパイラ命令並列化方式の動作を説明するための図であり、本発明のコンパイラ命

10

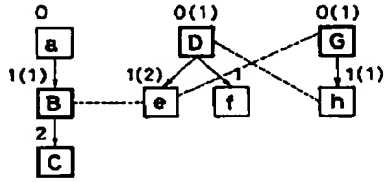
令並列化方式を使用しないと仮定した時のスケジューリング結果である出力コードの一例を示す図である。

【図６】本発明のコンパイラ命令並列化方式の第２の実施例の構成を示すブロック図である。

【符号の説明】

１　入力コード
２　フロントエンド
３，５　バックエンド
４　目的コード
６　スケジュール前目的コード
７，３２　コードスケジューリング部
３１　中間言語最適化部
３３　目的コード生成部
１００，２００　コンパイラ
３２１　依存・並列動作抑制グラフ生成部
３２２　パスレイテンシ・並列動作抑制数計算部
３２３　スケジューリング部

【図３】



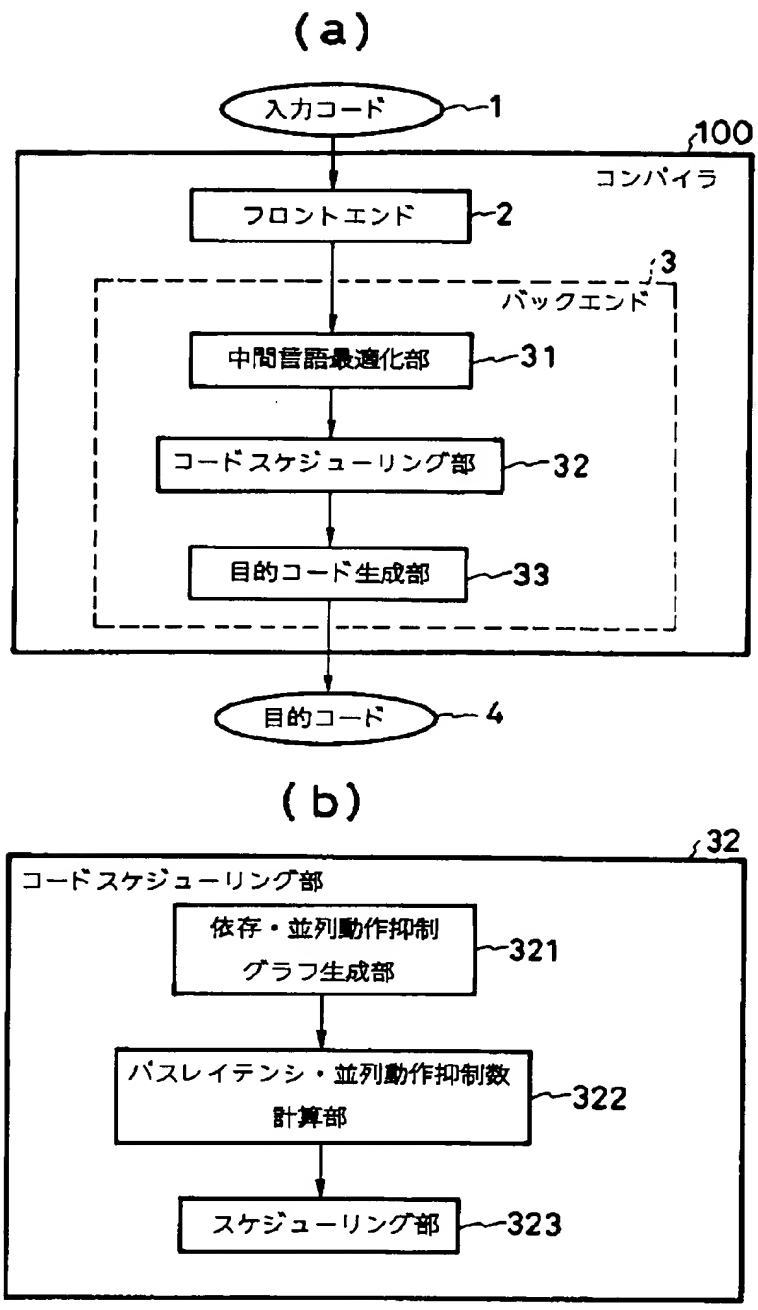
【図４】

演算ユニット ①	演算ユニット ②
a	D
f	G
h	B
e	C

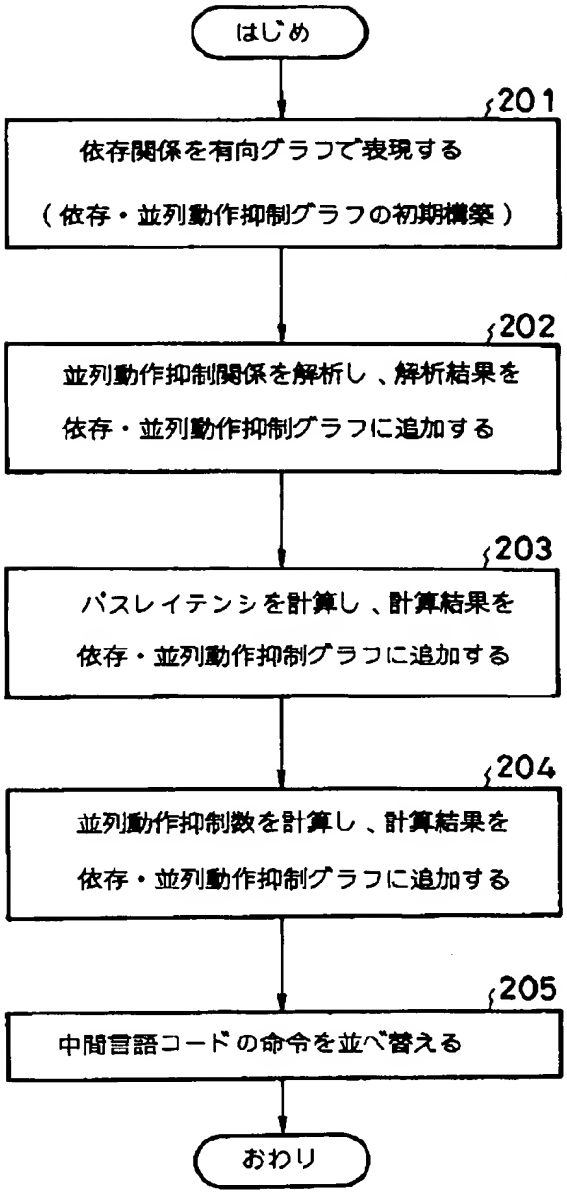
【図５】

演算ユニット ①	演算ユニット ②
	D
a	G
e	
f	B
h	C

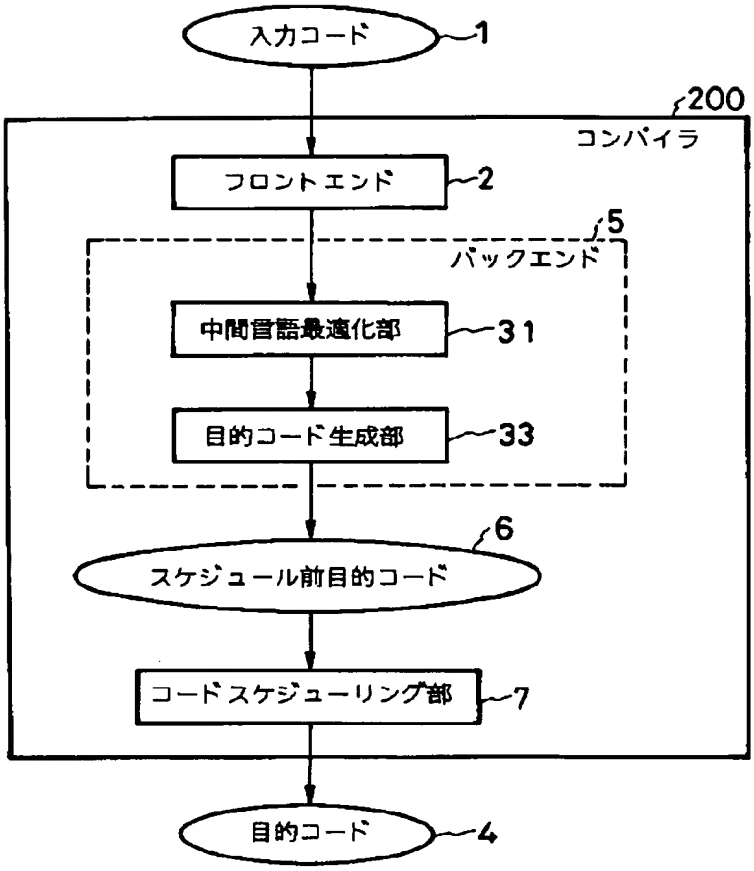
【図１】


(a)
入力コード
1
100
コンパイラ
フロントエンド
2
3
バックエンド
中間言語最適化部
31
コードスケジューリング部
32
目的コード生成部
33
目的コード
4
(b)
32
コードスケジューリング部
依存・並列動作抑制
グラフ生成部
321
パスレイテンシ・並列動作抑制数
計算部
322
スケジューリング部
323

【図２】


はじめ
201
依存関係を有向グラフで表現する
（依存・並列動作抑制グラフの初期構築）
202
並列動作抑制関係を解析し、解析結果を
依存・並列動作抑制グラフに追加する
203
パスレイテンシを計算し、計算結果を
依存・並列動作抑制グラフに追加する
204
並列動作抑制数を計算し、計算結果を
依存・並列動作抑制グラフに追加する
205
中間言語コードの命令を並べ替える
おわり

【図６】


入力コード
1
200
コンパイラ
フロントエンド
2
5
バックエンド
中間言語最適化部
31
目的コード生成部
33
6
スケジュール前目的コード
コードスケジューリング部
7
目的コード
4